SECURITY CLASSIFICATION OF THIS PAGE

(2)

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION AD-A210 009	1b RESTRICTIVE MARKINGS
	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
	5 MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 89-0978
6a NAME OF PERFORMING ORGANIZATION New York University	6b OFFICE SYMBOL (If applicable)
	7a NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research/NL
6c ADDRESS (City, State, and ZIP Code) 550 1st Avenue New York, NY 10016	7b ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448
8a NAME OF FUNDING / SPONSORING ORGANIZATION AFOSR	8b OFFICE SYMBOL (If applicable) NL
	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR-85-0341
8c ADDRESS (City, State, and ZIP Code) Building 410 Bolling AFB, DC 20332-6448	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO
61102F	2313	A5	

11 TITLE (Include Security Classification)

Automatic Construction of Polyhedral Surfaces from Voxel Representations

12 PERSONAL AUTHOR(S)
Eric Schwartz

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Reprint	FROM _____ TO _____		25

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

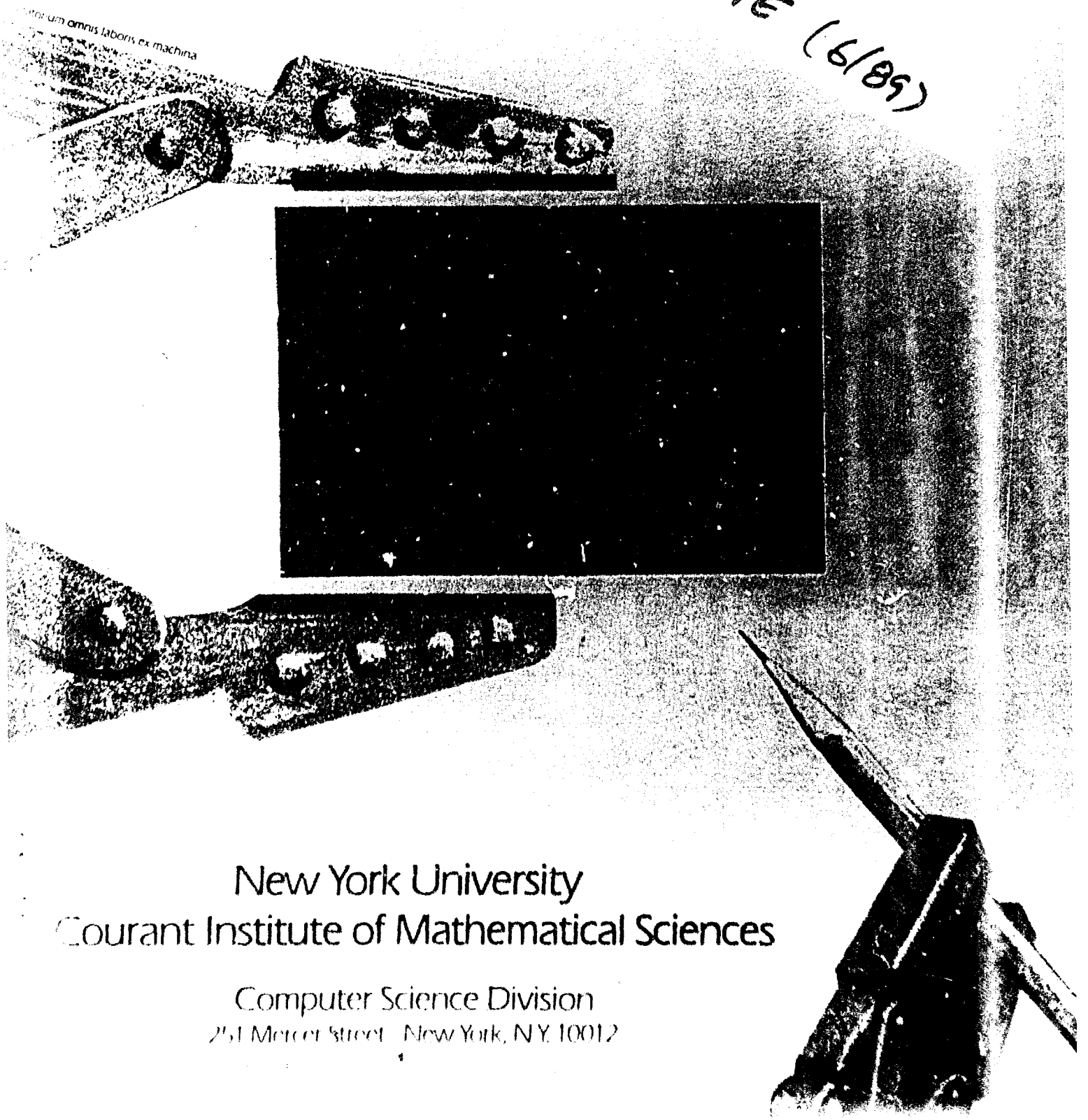
20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c OFFICE SYMBOL
John F. Tangney	(202) 767-5021	NL

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Robotics Research Technical Report



New York University
Courant Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street, New York, N.Y. 10012

Automatic Construction of Polyhedral Surfaces from Voxel Representations

by

Alan Shaw
Eric L. Schwartz

Technical Report No. 381
Robotics Report No. 158
June, 1988

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by AFOSR 85-0341, System Development Foundation and the Nathan S. Kline Psychiatric Research Center.

Automatic construction of polyhedral surfaces from voxel representations

†Alan Shaw

‡ †Eric L. Schwartz

†Computational Neuroscience Laboratories
New York University School of Medicine
Department of Psychiatry
550 First Avenue
New York, N.Y. 10016

‡Dept. of Computer Science
Courant Institute of Mathematical Sciences
New York University
715 Broadway
New York 10003 N.Y.

ABSTRACT

Various applications require triangulations, or polyhedral representations, of surfaces which are represented as serial sections. Heuristic methods are in common use to triangulate such data. These methods work well on segments of "generalized cylinder," i.e., runs of sections containing single loops, but they often fail when attempting to process highly convoluted surfaces. This is because the topology of the sections changes when a critical point of the surface is encountered. In this paper we use the equivalent of the full adjacency graph of the surface, provided by a voxel model, to classify the changes in topology of the sections of the surface, and thereby guide the triangulation process. For a voxel surface which is a discrete sampling of a smooth manifold in general position, we are able to exhaustively classify the small set of possible topological changes in the sections of the surface; we then deal with these cases exhaustively. To the best of our knowledge, this is the first description of an algorithm which can in theory and practice triangulate surfaces as complex as that of a brain, from serial sections, without human interaction.

May 19, 1988

supported by AFOSR 85-0341, System Development Foundation and the Nathan S. Kline Psychiatric Research Center

Automatic construction of polyhedral surfaces from voxel representations

†Alan Shaw

‡ †Eric L. Schwartz

†Computational Neuroscience Laboratories
New York University School of Medicine
Department of Psychiatry
550 First Avenue
New York, N.Y. 10016

‡Dept. of Computer Science
Courant Institute of Mathematical Sciences
New York University
715 Broadway
New York 10003 N.Y.

1. Introduction

Polyhedral representation of surfaces is a common requirement in computer graphics, CAD, numerical analysis, and biomedical applications. Commonly, the faces of the polyhedral model are triangles, and the modeling process is called triangulation. In biomedical applications, the surfaces are often represented as "serial sections," that is, by intersection of a series of parallel planes with the surface. These serial sections might be produced physically, by cutting with a microtome, or digitally, via tomographic reconstruction. The intersection of the surface with each cutting plane forms one or more closed curves, denoted "contours."

A variety of heuristics are in use to triangulate data represented as a sequence of contours of the surface. However, these heuristic approaches often fail when presented with complex surfaces (e.g. non-convex surfaces, surfaces containing saddle points and other critical points, etc.). Moreover, it is quite difficult to produce a triangulation of a complex surface with human interaction, because complex topological events ocurring in the sectional view must be visualized in 3-D in order to guide the triangulation process. In our experience, it has required several days of work to triangulate an object such as primate visual cortex interactively. Worse still, it is difficult to verify the correctness of the polygonal model.

In the present work, we present an implemented algorithm which automatically triangulates surfaces represented as serial sections. We illustrate the algorithm with the triangulation of the posterior half of a monkey brain (neo-cortex). We assume that a volumetric (voxel) model of the object to be triangulated is available, since the algorithm

supported by AFOSR 85-0341, System Development Foundation and the Nathan S. Kline Psychiatric Research Center

relies on surface tracking of such a model to instruct the triangulation module. We also assume that the model is sampled densely enough to be tracked unambiguously, i.e. that the voxel size is small compared to the smallest surface curvatures which are meaningful in the model. This algorithm can be viewed as a means of transforming from a voxel or cuberille data structure [Artzy et al., 1981] to a polyhedral data structure.

It might be thought that if one had access to a voxel model of a surface then triangulation would be trivial, because it could be provided by connecting all neighboring voxels. But economical models of a surface are necessary in cases where extensive numerical computation must be performed; the data compression offered by a sampled polyhedral model as compared to the voxel model is often the difference between feasible and infeasible computational burdens. For example, in our application, we desire to "flatten" or "unroll" the surface of the cortex. We have described algorithms (Schwartz et al., 1987c, Wolfson and Schwartz, 1988) that are capable of flattening visual cortex with minimal error, but these algorithms are numerically intensive. Our polyhedral model (Figure 2) is composed of triangles whose average surface area is about .5 mm^2, while our voxel models are composed of voxels whose faces are $40\mu^2$. There are about four orders of magnitude difference in size. Thus, a trivial polyhedral model constructed by brute force from a voxel model would be unsatisfactory. Similar arguments could be made in the context of CAD, CAM, motion planning, finite-element analysis, and other application domains. These applications require an economical surface representation, and would be overwhelmed with a voxel representation.

Conversely, voxel models do provide some unique advantages over polyhedral models. Powerful methods of voxel surface tracking are known [Artzy et al., 1981] which provide a full characterization of the surface connectivity. Moreover, if an application requires texture mapping (Perlin, 1985, Blinn and Newell, 1976) then a voxel structure is required. In our application, we wish to display high-resolution "maps" of neural architectures which are obtained from serial sectional data of primate visual cortex. (see Figure 3). This can only be done with a voxel model.

We have a need for easy interconversion between voxel and polygonal data formats. We accomplish conversion from polygonal to voxel format by scan conversion. But conversion from voxel to polygonal format is nontrivial, and is the subject of this paper.

2. Heuristic triangulation algorithms for surfaces

There are many algorithms which are capable of triangulating generalized cylindrical surfaces[Keppel, 1975, Fuchs et al., 1977, Christiansen and Stephenson, 1983, Cavendish et al., 1983], and others which attempt to deal with more general three-dimensional structures [Faugeras et al., 1984, Boissonnat, 1984, Anjyo et al., 1987, Zyda et al, 1987]. In our experience, none of these algorithms is sufficient to succesfully model a complex surface, such as that of primate cortex, without human supervision. Part of this problem is that even for a simply connected (in 3D) surface, contours might merge, split, be "born", "die", or change their internal structure as one moves from section to section. Adjacent sections of a surface can manifest changes in topology or

connectivity even if the surface is simply connected and closed. Such changes are purely an artifact of the sectional representation of a surface (see Figure 3). A major part of the solution to triangulation in this context is thus the proper association of contours on one level with those on adjacent levels.

One of the difficulties of constructing polygon-mesh models is the use of sparsely spaced contours, which either do not accurately reflect the structure of the actual surface, or have been obtained from a data source which itself does not provide full knowledge of the surface. Algorithms that attempt triangulation with incomplete information (we call them "simple triangulators") must rely on heuristics or interaction with a human to accomplish their task. We assume complete information about the surface to be triangulated; thus our algorithm does not rely on heuristics but on exhaustive classification of topological transitions in serial sections.

A widely used example of a "simple triangulator" is the Mosaic program distributed with Movie.BYU. Mosaic associates a contour in one level with a contour in the next level if their bounding boxes overlap. This is sometimes correct, but the facility allowing the user to override these assumptions is crucial to the program's triangulation capability. And there is no facility for partial associations: contours may not be broken up and the "pieces" reassigned, which is often necessary.

The more general and sophisticated algorithm of [Anjyo et al., 1987] uses region growing to partition contours for partial associations; however the base regions from which growing is done are still determined by simple two-dimensional overlap of contours. This is some improvement in that the contours themselves, rather than their bounding boxes, are overlapped, but there is still the potential for failure. The authors state that their algorithm assumes

> "that a sort of cross-sections' coherency exists... we know that such coherency does not always exist. We are currently working on how to circumvent this restriction."

It seems obvious that a true automatic triangulator must make use of the *adjacency* properties of the surface as it progresses from level to level. Our algorithm uses the adjacency graph of a voxel surface to perform an automatic triangulation.

3. Topological structure of serial sections of a surface

We begin by restricting the class of surfaces to be triangulated to those which can be viewed as discrete samples of some orientable differentiable manifold. This is a sufficiently broad class of voxel surfaces to include most natural objects of interest, and in particular, the surfaces of brains. We can then apply classification theorems from differential topology to enumerate the topological changes which can occur in the sequence of sections of a surface.

Intuitively, a differentiable manifold is a smooth surface which is locally like Euclidean space in that it admits of an atlas of local coordinate systems (differentiable mappings of overlapping open sets of the surface to overlapping open subsets of R^2).

Our motivation is to be able to discuss local minima of the surface, and to cite some powerful classification theorems which relate local minima to the kinds of topological transformations of serial sections which are at the root of the difficulty of the triangulation problem. (See [Wallace, 1968] for a more precise definition.)

The essential difficulty in triangulating a surface represented as serial sections arises from topological changes in the surface contours encountered during serial traversal of the sections. These changes can be localized at *critical points* of the surface. A critical point of a differentiable manifold occurs when the partial derivatives of the z coordinate or height function with respect to the local coordinate functions vanish [Wallace, 1968]. Geometrically, this corresponds to a situation in which the tangent plane to the manifold is "horizontal." This occurs at maxima, minima, and saddle points of the manifold. A section containing a critical point will be denoted a *critical level*.

As a critical point is passed in the sectional traversal, a single contour can split into several disjoint contours, or several disjoint contours can merge into one. These topological changes at critical levels incapacitate existing algorithms for triangulating from serial sections, which become confused when a contour that is being tracked suddenly splits into multiple contours, or vice versa.

As an example of a sectioned differentiable manifold, consider the surface of a doughnut standing on edge, and the serial sections of this object by horizontal planes(Figure 0). The following topological transformations occur. First, a point appears, the first critical point. We call this a "birth" event. At this level the section consists of a single point, which immediately opens into a loop as we ascend. A series of loops is produced until a second critical point of the doughnut is reached. At this point, the loop becomes a "figure eight", and splits into two disjoint loops. We call this event "splitting." Then, at the third critical point, the loops fuse back into a "figure eight". We call this a "merge" event. The figure eight immediately becomes a simple loop. Finally, the single loops shrink down to a point and disappear at the fourth and final critical point (a "death" event).

For the embedding of the torus implied by this sectioning, there are four isolated critical points. It can be proven that any differentiable 2-manifold can be embedded in Euclidan 3-space in such a way that its critical points are finite in number, and no more than one occurs on a given level. Also, it can be shown that the only topological changes which can occur are the "birth" , "death", "splitting" and "merging" events illustrated for the torus[1].

Each topological change is accompanied by an Euler transition, that is, a change in the Euler number of a contour. (The Euler number of a component is one minus the number of holes; the number of holes, in turn, is "one less than the number of connected components in the complement of the figure." [Duda and Hart, 1973]). These changes

[1] There is a fifth event, which is "twisting" (see [Wallace, 1968]), but this can only occur for nonorientable surfaces such as the Klein bottle, which we exclude from consideration in the present algorithm.

are exemplified by the figure eight-to-loop and point-to-loop transitions in the torus. On a manifold, an Euler transition can occur nowhere but at a critical point. As can be seen from the example, the critical level differs from one of its two adjacent levels by a change in the number of contours, and it differs from the other adjacent level by an Euler transition.

Since the critical points are finite and isolated, and the kinds of topological transitions that they cause are only four in number, an algorithm can be constructed which deals with each of these possible cases. That is, if we can locate the "critical points" in the discrete sampling of our voxel surface, we can build a data structure which represents our original set of serial sections as a series of "critical points," joined by segments of generalized cylinders. Using this syntactic description of the surface, it is then possible to write a series of instructions, or program, for a simple triangulator to complete the triangulation.

4. Description of the algorithm

Having stated, in terms of critical points and critical levels of a differentiable manifold, the problems which a triangulation algorithm must solve, we now turn to a discrete approximation to the manifold, or surface, which we wish to triangulate.

Consider a series of voxel-thick contours obtained by intersecting the surface with parallel planes. These contours are connected components of voxels (see Figure 3). Contours on adjacent levels have adjacency relationships, which allow us to "parse" the surface into *generalized cylinders*, which a simple triangulator can process. (A generalized cylinder is a run of simple loops. It contains no critical points, and represents a (distorted) sausage-like object [Binford, 1971]). In our experience, even complex surfaces such as cortex consist of relatively few critical levels with long runs of generalized cylinder between them. We find the critical points where the four types of transitions — "birth", "death", "merging", or "splitting" — occur, and write a program or script for the simple triangulator. This script contains the commands a human would normally be required to provide to instruct the simple triangulator on the proper association of contours around the critical levels. We use the Mosaic program of Movie.BYU for the simple triangulator stage of the process. (The "compiler approach" which we follow makes it easy to substitute other simple triangulators. In effect, we construct a high-level syntactic description of the surface in terms of critical points and generalized cylinders, and then write a program in a low-level language (such as a script for Mosaic) to effect the triangulation. We summarize the algorithm as follows:

```
Find and label connected components of sections
Detect critical levels
On each critical level:
    Find adjacency sets in each contour which has multiple nei
    Grow adjacency sets to find fences
    Find intersection of fences to yield critical points
    Thin, track, and subsample contours
Construct program for simple triangulator
Execute simple triangulator
```

Find and label connected components of sections

Two voxels are *edge-adjacent* if they share at least a common edge. Two contours on adjacent sections are *neighbors* if there is at least one voxel edge-adjacency between them. See Figure 4.

Figure 3 shows examples of contours. The neighbor relation between contours on successive sections allows an unambiguous tracking of contours across sections until a critical level is encountered. Each contour is given a unique label, and this label is propagated through the sections. Thus, a generalized cylinder is identified as a series of contours with the same label. These can be triangulated easily by the simple triangulator.

Detect critical levels

A critical level is identified when a contour has either no neighbors or more than one neighbor on an adjacent section. Figure 3 shows several examples of critical levels, which are characterized by a change in 2D connectivity of the sectional description of the surface. These critical levels are detected, and a simple syntactical description is constructed to describe the events which have occurred at each critical point.

The following is an example of the syntactic description produced for the brain illustrated in the present paper. There are a total of 13 critical levels in a run of 290 sections representing much of the posterior pole of the brain of a macaque monkey.

```
 21:   1 birth
 90:   2 birth
113:   1 + 2 merge into 2
115:   2 splits into 2 + 3
116:   3 death
151:   4 birth
151:   5 birth
153:   4 + 5 merge into 5
163:   2 + 5 merge into 5
196:   6 birth
231:   5 + 6 merge into 6
248:   7 birth
295:   6 + 7 merge into 7
310:   7 death
```

Find adjacency sets in each contour which has multiple neighbors

When a contour has multiple neighbors on an adjacent section, it is necessary to partition it into regions which will each be associated with a unique neighbor. We begin by finding parts of the contour that are easily associated with a particular neighbor. If two contours A and B are neighbors, the *adjacency set* of B in A is the set of voxels in A that are edge-adjacent to voxels in B.

Figure 4 shows an example of several adjacency sets.

Grow adjacency sets to find fences

Region growing is the accretion of pixels to a component in all directions simultaneously, subject to certain limiting criteria. Application of region growing to a set of adjacency sets within a contour in parallel, adding only contour pixels, and terminating when all contour pixels are used, results in a generalized Voronoi diagram[2] which represents the boundaries among their grown regions. Each of these regions will be output as a separate contour for connection to one of the neighbors. We call the boundary a *fence*. Fences are shown in Figure 5 and Figure 6.

Find intersection of fences to yield critical points

The fence is also used to localize the transition at a single *common node*, a representative pixel which will be present in the description of each of the contours that we output as data for the simple triangulator. We identify the common node with the

[2] "The Voronoi diagram of a finite set S of points in the plane is a partition of the plane so that each region of the partition is the locus of points which are closer to one member of S than to any other member." [Preparata and Shamos, 1985, p. 234]. We generalize the concept by substituting a set consisting of other geometric objects, namely adjacency sets, for the set S of points.

critical point in the continuous model, the point common to all of the contours involved at the critical level.

The common node is selected to lie on the fence, or if there are neighbor contours on both adjacent sections (see Figure 6), on the intersection of the two fences. A reasonably central point is found by a symmetrical thinning operation.

Having found critical points to represent sets of pixels involved in transitions, it is a simple matter to link together the respective generalized cylinders (Figure 7).

Thin, track, and subsample contours

To "thin" a 2-D connected component is to remove boundary pixels as long as their removal does not change the Euler number of the component [Davies and Plummer, 1981]. Thinning produces a "skeleton" of the connected component, which we call an 8-contour. Tracking produces an ordered list of the pixels in an 8-contour, which is subsampled (Figure 8) to provide data for the simple triangulator.

Figures 7A and 7B show contours before and after our thinning operation. In the thinning operation, the common node is protected from removal, and adjacent pixels that are separated by a fence segment are not considered connected. This use of the fences ensures that thinning preserves a distinct path into and out of the common node each time it will be traversed in the tracking operation. Each path into or out of the common node is separated from the others by a fence. The common node itself is temporarily represented as a small "blob" of pixels large enough to straddle the fences that pass through it.

The tracking operation shows the crucial difference in the treatments of the contour with respect to the configurations of the adjacent sections. In the tracking operation, fences derived from the "other" adjacent section may be crossed when we get to a common node, and so we will be able to track from our path into the common node to an adjacent path out of the node without ever crossing our own path. The subsampling is done by the standard "polyline approximation to a curve" algorithm [Duda and Hart, 1973, p. 338].

Fences and common nodes are found where a contour has multiple neighbors on at least one of its two adjacent sections; on such contours, tracking and subsampling are done twice — once with respect to each adjacent section. The same set of nodes is produced both times, but grouped differently.

Thinning, tracking, and subsampling are applied not only at critical levels: levels are interpolated between critical levels in order to maintain fidelity to the segments of generalized cylinder. At present the interpolation is linear in z; a possible improvement would be to vary it with the longtitudinal curvature.

Construct program for simple triangulator

Using the syntactic description of the critical levels (Table) above, a script is written for a simple triangulator to produce the final triangulation of the surface, using the thinned and subsampled points obtained from the contours above. The triangulation of

the small piece of brain surface which has been illustrated in Figure 3 is shown in figure 9.

5. Discussion

We have described an algorithm which takes as input a voxel model of a surface, in the form of the intersections of this surface with a series of parallel two-dimensional voxel planes. It locates those sections in which a topological change occurs in the sectional view of the surface, parses the surface into segments of generalized cylinder separated by these critical levels, and then generates a program for another program (a simple triangulator) to execute, as well as the input data for the simple triangulator.

We demonstrate the correctness of this algorithm by citing several theorems of differential topology, which classify the critical points of a differentiable manifold and its behavior in the neighborhood of critical points.

Of principal importance is the embedding theorem, proven in [Wallace, 1968], which states that there is an embedding of a 2-manifold with boundary in Euclidean 3-space such that the critical points of the manifold are isolated, with no more than one critical point occurring on a given level.

It can also be shown that the sections of the 2-manifold are generally of the form of loops, which are diffeomorphic to a circle. The topological changes which can occur to these loops, as one moves through the stack of sectioning planes, are of the form of appearance, disappearance, merging, or splitting of loops (and twisting of loops, for nonorientable surfaces).

This analysis indicates that only a small number of events can occur: critical points are isolated and finite, and they can only participate in the four types of topological changes indicated above.

We have implemented an algorithm which makes use of surface tracking to find these critical points, in the discrete sampling provided by a voxel surface, and which determines how the topology changes across these critical levels.

It is important to point out, however, that we generally take the embedding to be the one implicitly provided by the original choice of sections. In our case, this is the plane of the cutting of the original brain. It is, of course, possible to produce other planes of section, since we produce the sections digitally. In practice, the only property of the critical points which we rely upon is that they are finite and separated. Consider, again, the case of the torus. There is an embedding of the torus for which the critical points are not separated and are infinite. This occurs when the torus is flat on the plane of section. Our algorithm will not handle this case because the "common node" in this case is really a non-simply connected circle of points, i.e. a set of non-isolated critical points. However, the embeddings which have this problem are structurally unstable[Golubitsky and Guillemin]: any slight rotation of the torus will revert to the generic situation in which the torus has four isolated critical points. If a non-generic situation occurs in practice, say, because one insisted on triangulating a doughnut flat on a table, then our algorithm would report the problem, and we would perform a small random rotation of the plane of

section and begin again. In practice, we have never encountered such a situation.

An alternative to this method would be to implement a procedure to find the desired generic embedding. Methods to find coordinate transformations which would provide this embedding are outlined in [Wallace, 1968]. We have developed an algorithm to embed a surface in general position, i.e. such that its critical points are isolated and non-denegerate, but so far we have not found a practical motivation to use this algorithm.

Generalization of the method to triangulating from sparse sections

The algorithm outlined in this paper requires detailed knowledge of the surface connectivity of the surface to be triangulated. Often, this is unavailable. In the case where such information is unavailable, and where there are critical levels in between the levels supplied by the data, it would seem that the problem is ill-posed. However, the methods and analysis of the present paper could be used to produce a heuristic "geometry parser" which would attempt to provide a sensible linkage of contours which are "splitting" and "merging", based on the syntactic and semantic metaphors of the present paper. We are currently working on such an algorithm, which can be evaluated in comparison with the many existing heuristic algorithms.

In situations where there is sufficient detailed knowlege of surface connectivity to disambiguate the topological changes which occur in the neighborhood of critical points, the algorithm of the present paper will provide a correct and automatic triangulation of that surface.

6. Acknowledgement

We are indebted to Alan Rojer for invaluable editorial assistance and discussion and to William Light for discussions during the early stages of this work.

References

Anjyo.

Anjyo, Ken-ichi, Toshio Ochi, Yoshiaki Usami, and Yasumasa Kawashima, "A practical method of constructing surfaces in three-dimensional digitized space," *The Visual Computer*, vol. 3 no. 1, pp. 4-12, 1987.

Artzy.Artzy, E., G. Frieder, and G. T. Herman, "The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm," *Computer Graphics and Image Processing*, vol. 15, pp. 1-24, 1981.

Binford,.

Binford, T. O., "Visual Perception by Computers," *Proc. IEEE Conf. Systems and Control*, IEEE, Miami, Fla., 1971.

Blinn.Blinn, J. F. and Martin E. Newell, "Texture and Reflection in Computer Generated Images," *Communications of the ACM*, vol. 19, October 1976.

Boissonnat,.

Boissonnat, J. D., "Geometric structures for three-dimensional shape representation," *ACM Trans. on Graphics*, vol. 3 No. 4, pp. 266-286, 1984. Cavendish, James C., David A. Field, and William H. Frey, "An approach to automatic three dimensional finite element mesh generation," *General Motors Tech. Rep.*, vol. GMR-4533, 1983.

Christiansen.

Christiansen, Hank and Mike Stephenson, *MOVIE.BYU*, University Press, Brigham Young University, 1983.

Davies.

Davies, E. R. and A. P. N. Plummer, "Thinning algorithms: a critique and a new methodology," *Pattern Recognition*, vol. 14, pp. 53-63, 1981.

Duda.Duda, Richard O. and Peter E. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

Faugeras.

Faugeras, O. D., M. Hebert, D. Mussi, and J. D. Boissonnat, "Polyhedral approximation of 3-D objects without holes," *Computer Vision, Graphics, and Image Processing*, vol. 25, pp. 169-183, 1984.

Fuchs.

Fuchs, H., Z. M. Kedem, and S. P. Uselton, "Optimal Surface Reconstruction from Planar Contours," *Communications of the ACM*, vol. 20, pp. 693-702, 1977.

Golubitsky.

Golubitsky, Martin and Victor Guillemin, *Stable mappings and their singularities*, Springer-Verlag, New York-Heidelberg-Berlin, 1973.

Keppel,.

Keppel, E., "Approximating complex surfaces by triangulation of contour lines," *IBM J. Res. Devel.*, vol. January, pp. 2-11, IBM, Yorktown Heights, 1975.

Perlin,.

Perlin, Ken, "An Image Synthesizer," *Computer Graphics*, vol. 19, pp. 287-296, 1985.

Preparata.

Preparata, Franco P. and Michael Ian Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

Schwartz.

Schwartz, Eric L., Alan Shaw, and Estarose Wolfson, "The generalized mapmaker's problem: optimal flattening of polyhedral surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear 1988.

Wallace,.

Wallace, Andrew H., in *Differential Topology*, W. A. Benjamin, New York, 1968.

Wolfson.

Wolfson, Estarose and Eric L. Schwartz, "Computing minimal distances on arbitrary polyhedral surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. (In press), 1987.

Zyda.Zyda, Michael J., Allan R. Jones, and Patrick G. Hogan, "Surface construction from plane contours," *Comp. & Graphics*, vol. 11, pp. 393-408, 1987.

Figure captions

Figure 0.

This figure shows a torus, and the location of its four critical points: a maximum, two saddle points, and a minimum. Below the first saddle point, the sections of the torus are single loops. Above this critical section, whose contour resembles a "figure-eight", the contours split into two sets of loops/section. The torus can be represented in terms of its critical points, and the generalized cylinders which lie between critical levels. In general position, any differentiable manifold has a finite, separated set of critical points, so that the situation illustrated in this figure is generic for the kinds of surface which we wish to triangulate.

Figure 1.

This figure shows a surface composed of voxels. It is a single-voxel layer of the back half (occipital pole) of a monkey brain, in which a zebra-skin like pattern of metabolic activity appears. This pattern (ocular dominance columns) represents the terminations of the left (dark) and right (light) eyes in the brain. This structure was produced by repeatedly surface-tracking the entire brain. Each "surface" is saved, and the process repeated, resulting in some 25 voxel surfaces such as this one, which is located in the center of the cortex (layer IV). The size of these voxels is 40 μ x 40 μ x 40 μ, and the figure is composed of 414,211 voxels. Note that the voxel surface is independent of the gray-scale detail, which we have shown here merely for graphic interest.

Figure 2.

This figure shows a polyhedral model of the same brain represented in figure 1, in terms of triangular surface patches, of typical size about .5 mm^2. This triangulation is the final output of the algorithm of the present paper, that is, this triangulation was produced completely automatically, with no human interaction or decision. There are 1776 triangles in this model.

Figure 3.

On top, we show examples of the individual serial (coronal) sections of monkey brain, used to make figures 1 and 2. The "zebra-skin" pattern of the ocular dominance columns is seen in cross section here. On the bottom are shown the contours associated with the voxel surface shown in figure 1, for these five sections. Note that the sections have been thresholded to a binary image prior to obtaining the contours, since the gray scale detail is irrelevant to the present demonstration. These contours are produced by intersecting the voxel surface with parallel planes Note that the contours can have "thickness": they are merely connected components, not necessarily 8-contours.

Section 150 is the end of a run of uneventful sections, comprising a *generalized cylinder*. There are two transitions in section 151: two new contours suddenly appear (which are pieces of cortex "wrapped around" in 3D, but continuous and connected with the surface of cortex represented in 150). These are birth transitions.

In section 153 there is another transition, of the split-merge type; the two contours become one.

Figure 4.
top center: a contour of section 153.
top left and right: its neighbors on the previous and succeeding sections.
bottom: the *adjacency sets* in our contour of its neighbors. These are the sets of voxels in the middle contour that are actually edge-adjacent to voxels in the neighbor contours.

Figure 5

The *grow* operation, applied to the adjacency sets shown in figure 4. The first frame on top left shows the adjacency sets of the contour's neighbors in section 152; top right shows those of the neighbors in section 154. Every third step of the grow is shown. bottom: the contour pixels shaded according to their

rank in the grow. The darkest pixels are the adjacency set; the lightest pixels lie along the fence. On the left, there is a merge/split transition (see the text), and the fence is also visible in the last step of the grow as the boundary between the two differently-shaded growth regions. On the right, there is an Euler transition (see the text), and the fence is the limit of the thread of ungrown pixels connecting the two holes.

Figure 6.
Top: fully grown regions in the contour, with the fences shown. We show the fences as chains of pixels; actually they are boundaries between sets of pixels.
Center: the fences of the transitions on either side of the section, shown individually and then superimposed.
Bottom: The intersection of the two fences yields the *common node* which we will use to join the generalized cylinders obtained by joining the points of this countour to simple contours on either side.

Figure 7A.
left top and bottom: our contour's neighbors on the previous section
right center: our contour's neighbor on the following section
center column: three views of our contour. Next to each neighbor we see the grown region of the neighbor in our contour.

Figure 7B.

Results of thinning as applied to all the regions in figure 7A. Note that the full set of loops of our contour obtained by thinning with respect to the previous section (center column, top and bottom pictures) is the same as that obtained with respect to the next section (center column, second picture).

Figure 8.
top: results of tracking and subsampling as applied to the thinned contours of the previous figure; previous section on left, our section in center, next section on right. Note that for triangulating to the previous section, we obtained two loops that happen to share a node; for triangulating to the next section we obtained one loop that happens to pass through a certain node twice. These versions both consist of the same set of points, pictured at top center.
bottom left: triangulation to the previous section as performed by Mosaic under our program's direction
bottom right: triangulating to the next section similarly Note: these pictures are not to the same scale.

Figure 9

Shaded view of the triangulation of the surface illustrated in Figure 3.

FIGURE 0

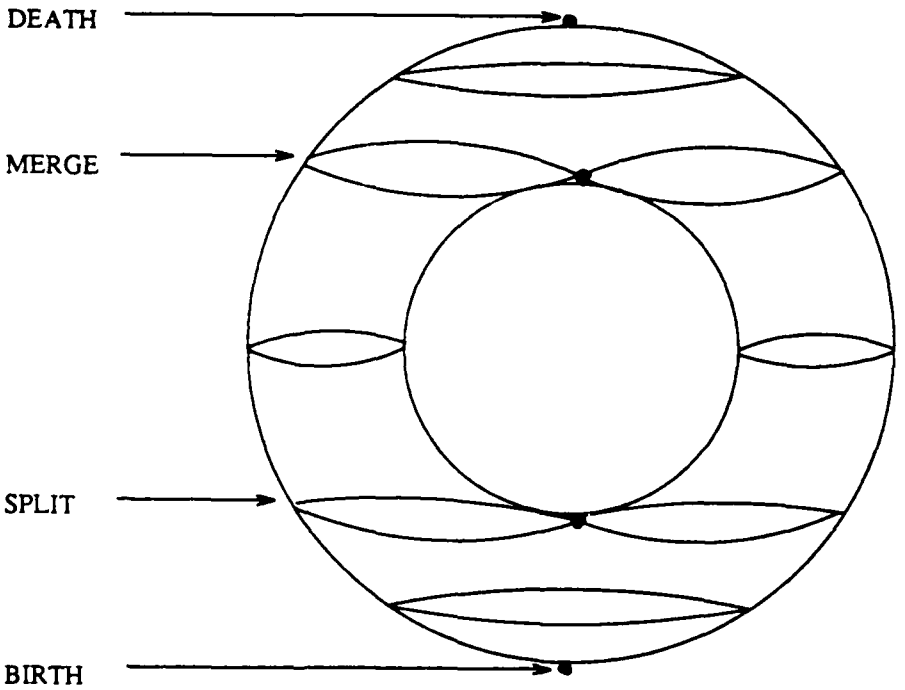


FIGURE 1

FIGURE 2

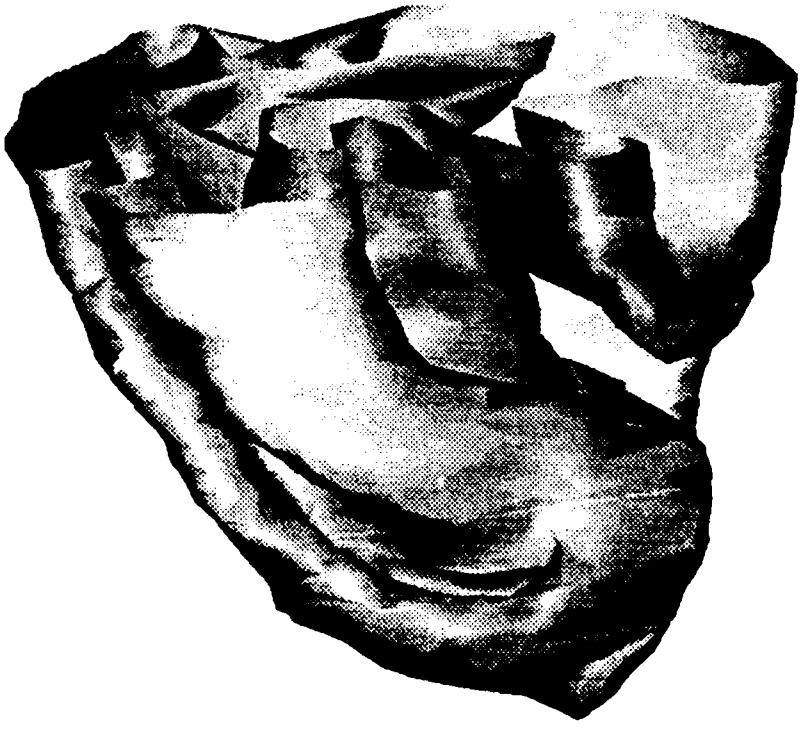


FIGURE 3

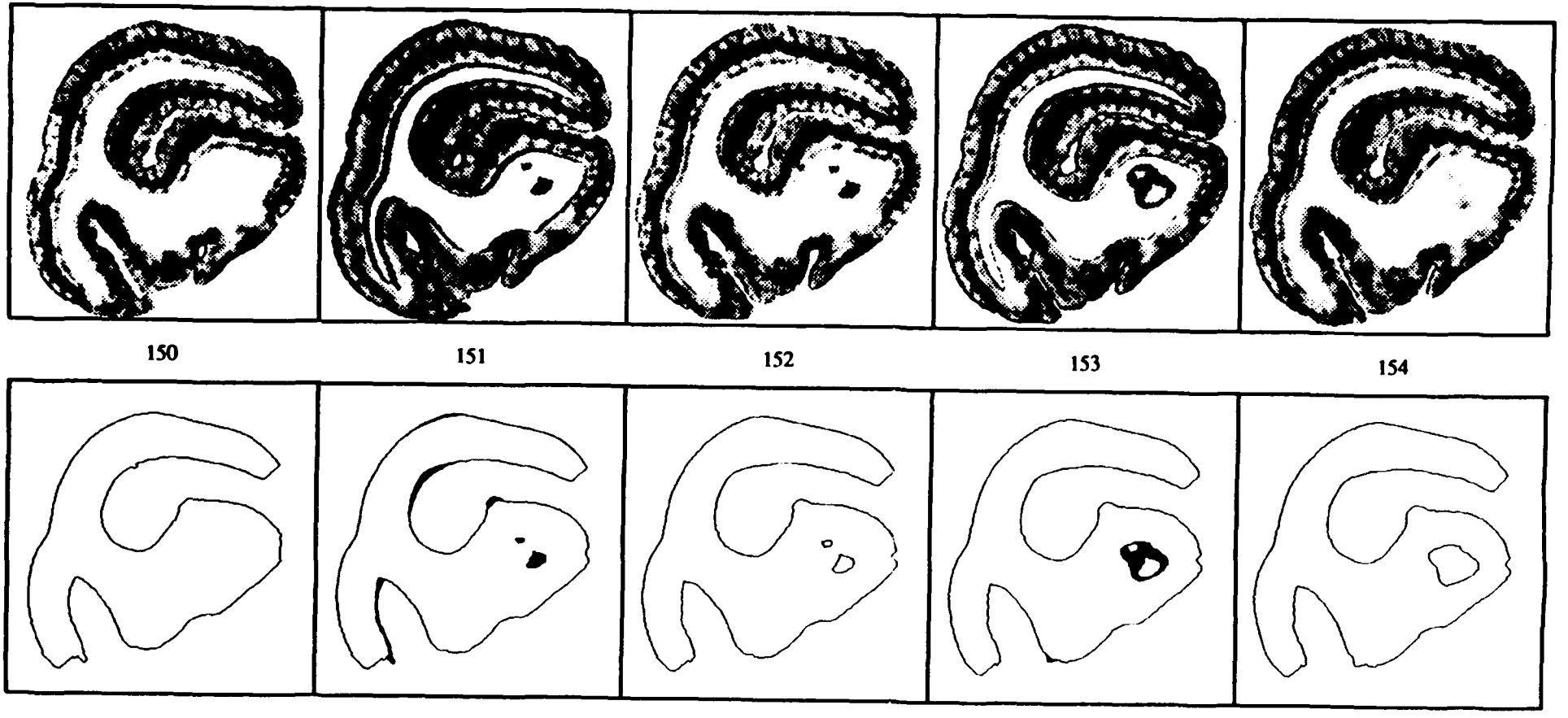
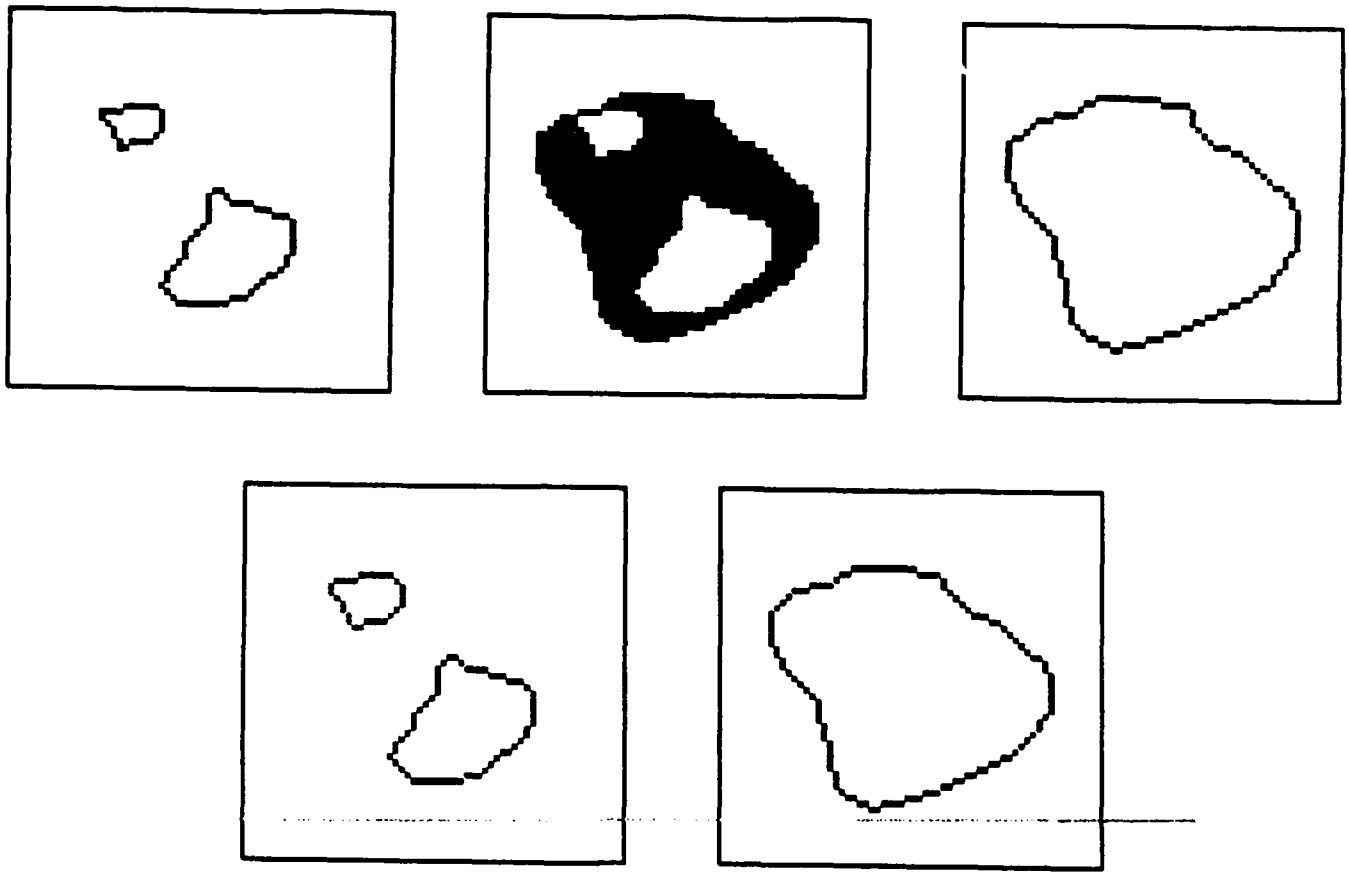


FIGURE 4



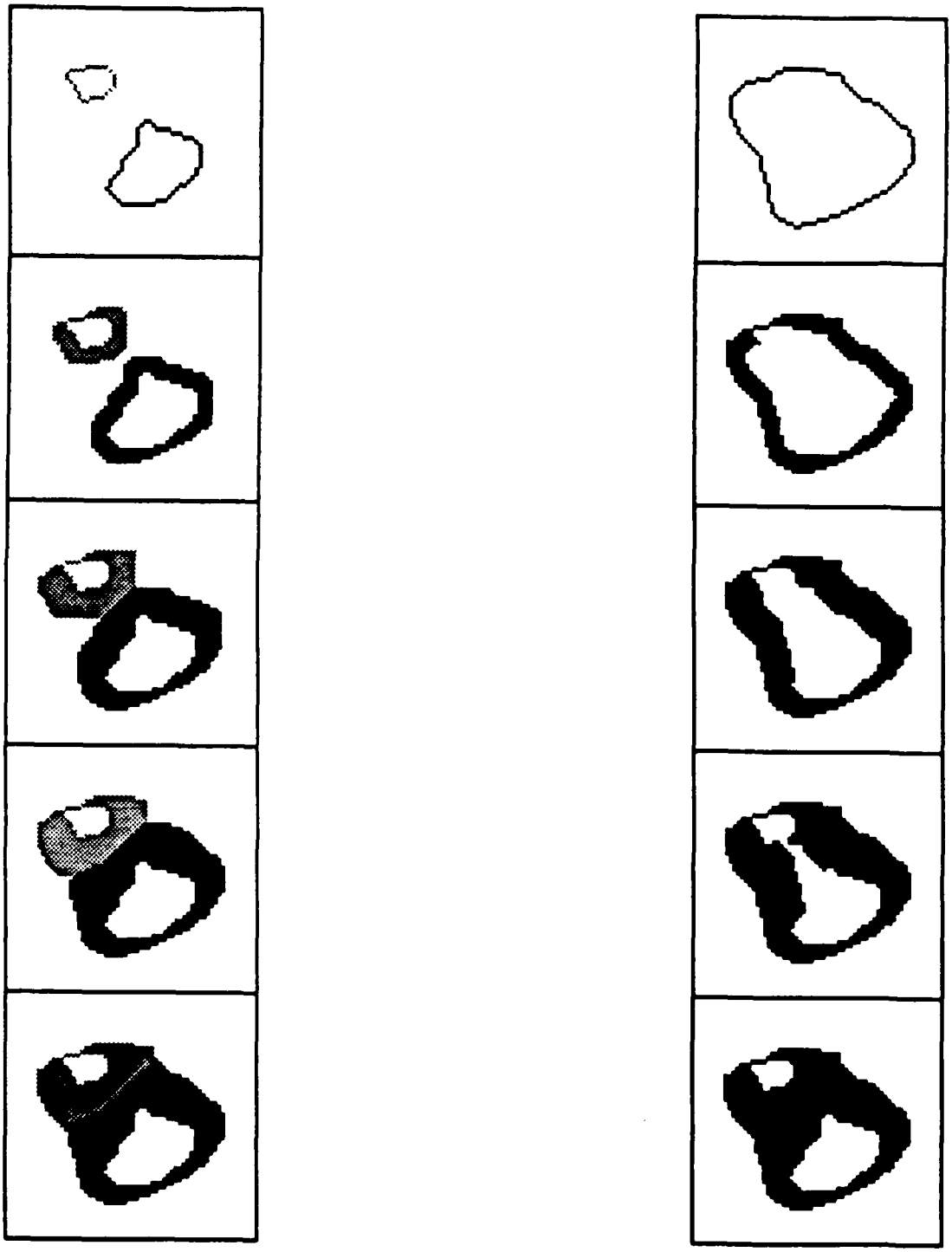


FIGURE 5

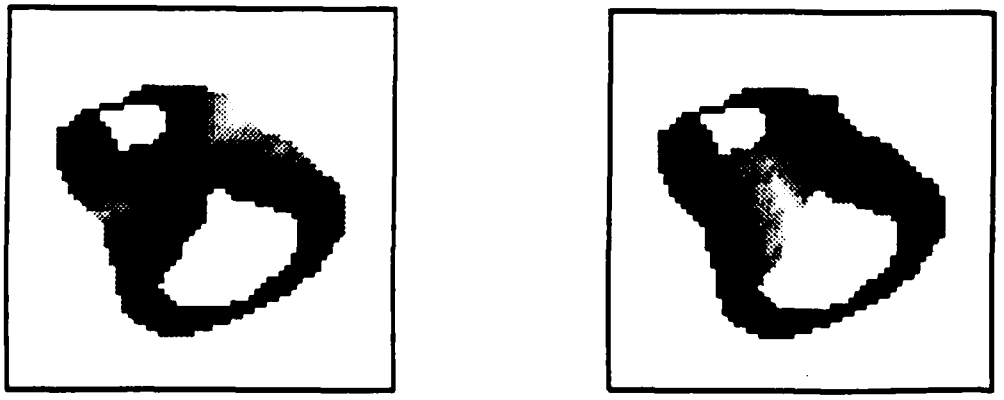


FIGURE 6

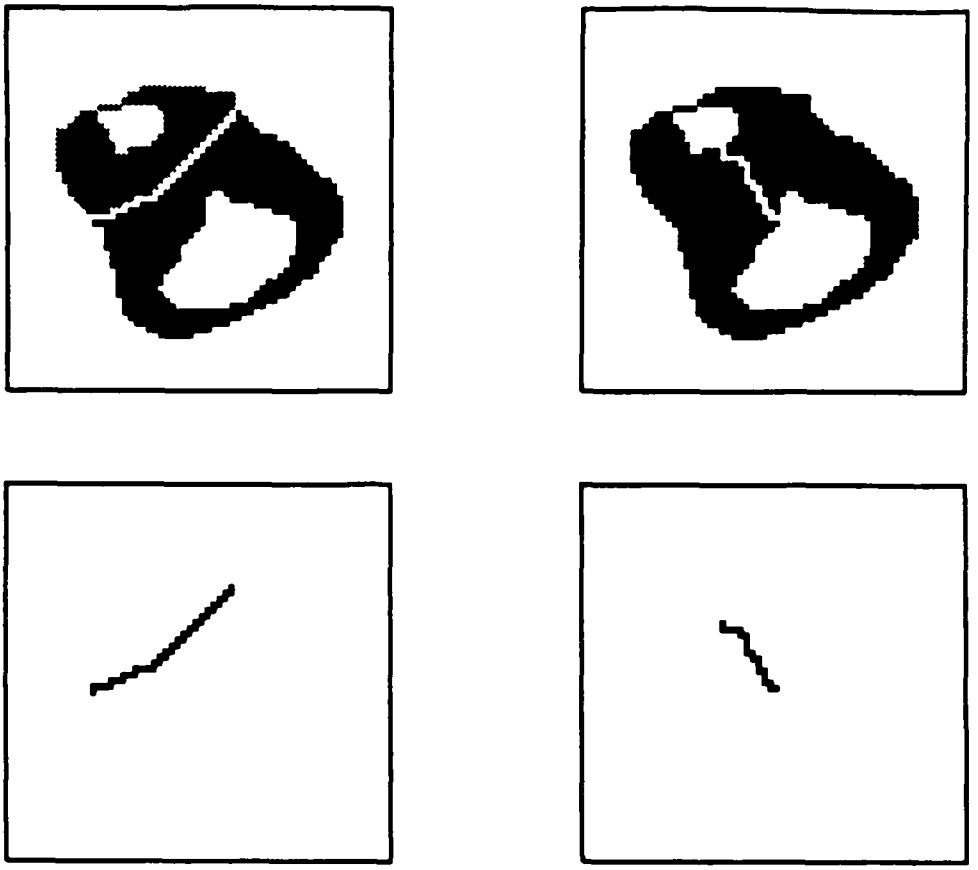


FIGURE 7A

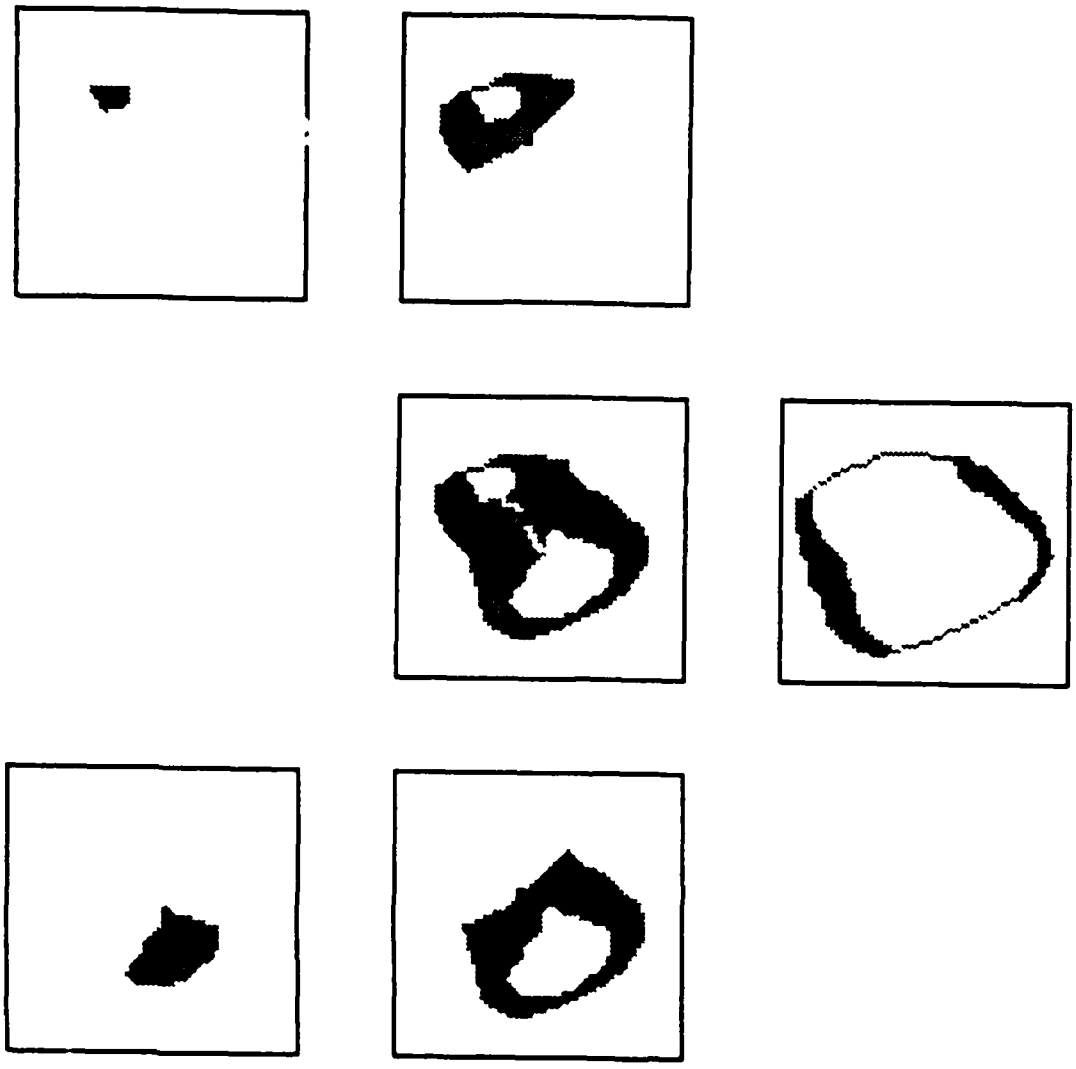


FIGURE 7B

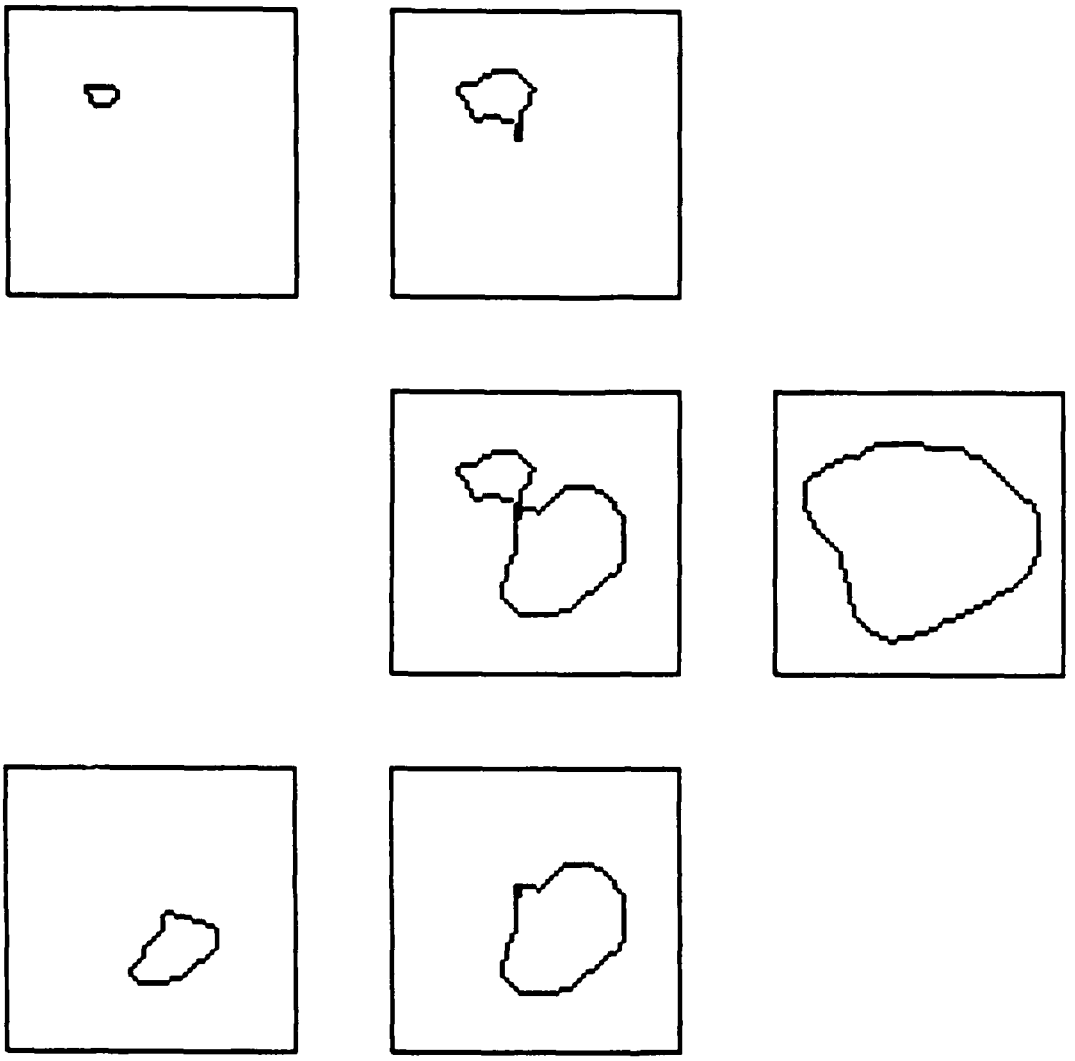


FIGURE 8

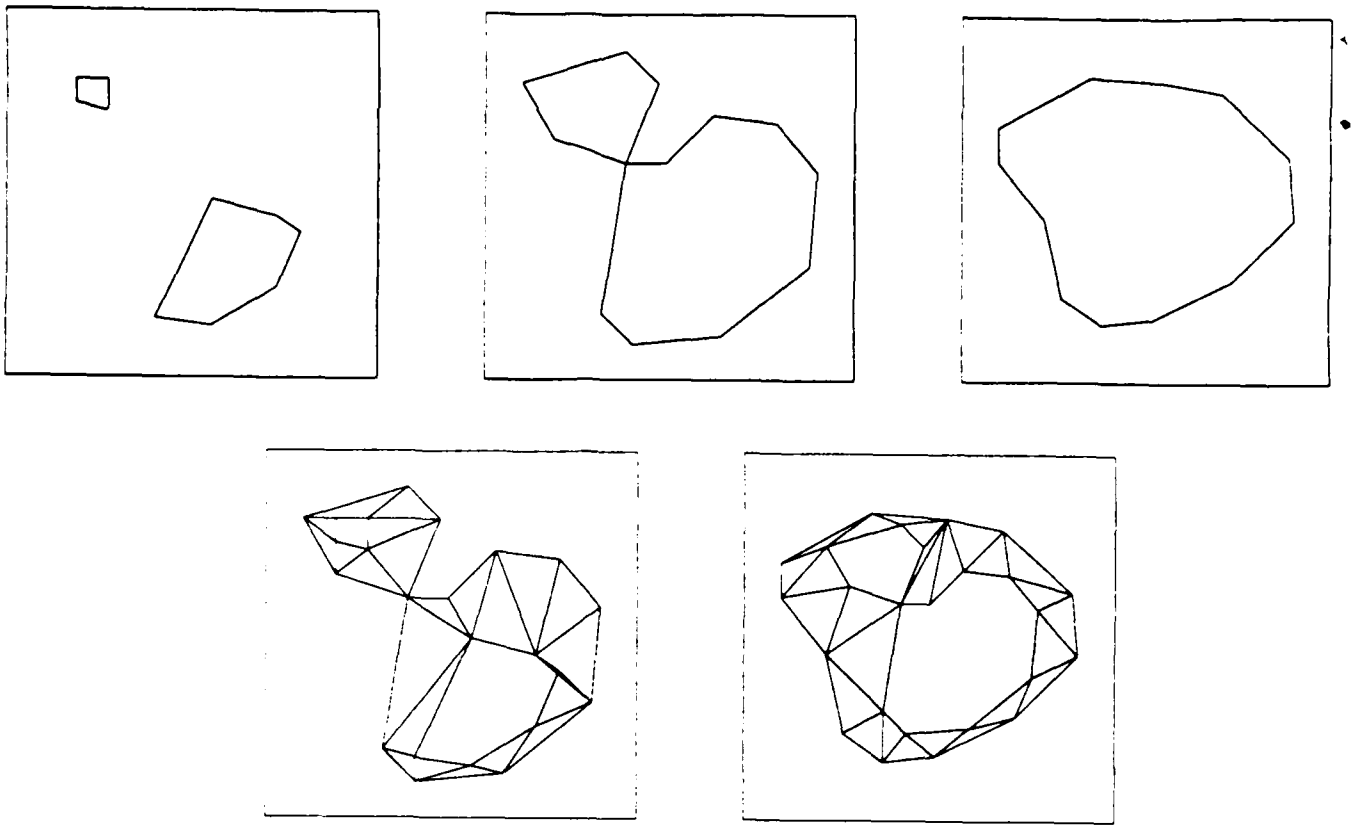


FIGURE 9